Robotica (2012) volume 30, pp. 363–378. © Cambridge University Press 2011
doi:10.1017/S0263574711000701

High-accuracy global localization filter for three-dimensional environments

Fernando Martín, Luis Moreno, Santiago Garrido and Dolores Blanco

Carlos III University, Madrid, Spain

(Received in Final Form: June 10, 2011; accepted June 9, 2011. First published online: July 11, 2011)

SUMMARY

The localization problem in mobile robotics can be define as the search of the robot's coordinates in a known environment. If there is no information about the initial location, we are talking about global localization. In this work, we have developed an algorithm that solves this problem in a three-dimensional (3D) environment using evolutionary computation concepts. The method has been called RELF-3D and has many features that make it very robust and reliable: thresholding and discarding mechanisms, different cost functions, effective convergence criteria, and so on. The resulting global localization module has been tested in numerous experiments and the most important improvement obtained is the accuracy of the method, allowing its application in manipulation tasks.

KEYWORDS: Mobile robots; Genetic algorithms; Global positioning systems; Optimization problems.

1. Introduction

There are many tasks in which an autonomous robot needs to know accurately where it is to carry them out successfully. The localization problem is crucial in robotics and can be define as the search of the robot's coordinates relative to its environment, assuming that it is provided by a map. For instance, it is required to execute a geometrical path successfully during navigation. We can distinguish between two different situations: the re-localization problem (the robot knows its initial position, at least approximately) and the global localization problem (no knowledge about the robot's initial position). The second case will be studied in this paper.

In our firs approaches, we tried to solve the referred problem in a two-dimensional (2D) map with an evolutive filte .[19] The algorithm developed was called Evolutionary Localization Filter (ELF) and solves the global localization problem in a satisfactory way. Due to the available sensors and the accuracy requirements of certain tasks, the extension to 3D environments and an improvement in the method were developed after that.[17] In our ELF approach it was assumed that the robot was in a 2D map, and three parameters were estimated, corresponding to the positions (x, y) and orientation (θ). We are now working on 3D maps, and four coordinates must be calculated, corresponding to the position (x, y, z), and horizontal orientation (θ). The degrees of freedom stated is 4 for simplicity (because our robot works in 4 DOF) but the robot could work in 6 DOF $(x, y, z, \psi, \theta, \phi)$ when the map is three dimensional. The current method can be easily expanded to 6 DOF without increasing the computational cost.

The new algorithm has been called Rejection Evolutionary Localization Filter in Three Dimensions (RELF-3D). A detailed explanation of its fina version and a complete study of last developments and experimental results are presented in this paper.

The RELF-3D method is based on the representation of the robot's location by a set of possible location estimates weighted by a fitnes function. The state is recursively estimated using a set of results selected according to the weight associated to each possible solution included in the set. The solutions set evolves in time to integrate the sensor information and the robot motion information. The adaptation engine of the RELF-3D method is based on an evolutive adaptation mechanism, which combines a stochastic gradient search together with a probabilistic search to fin the most promising pose candidates.

Our research group is working on the development of experimental platform MANFRED-V2 (Fig. 1). This robot has some built-in sensors, which have been used in this work.

The firs one is a laser range finde (SICK PLS), which provides us with 3D information about the environment. The original sensor measurements are bi-dimensional, but we have added a motor that lets it rotate up and down, being able to obtain measurements in three dimensions. In order to understand this, imagine that the laser rotation movement is like a pan-tilt camera. The maximum horizontal (tilt) resolution, which is the minimum distance between two consecutive measurements within a scan in two dimensions, is 0.25°, and the amplitude of a 2D scan is 190°. The vertical (pan) resolution, which is the minimum distance between two consecutive 2D scans, is 1° .

The 3D laser reading is used by our localization module to try to locate the robot in a familiar environment. A typical environment and a real 3D laser reading can be seen in Fig. 2. We have also implemented a simulated laser, which works in thirteen vertical scans separated by 5° degrees, and each vertical scan contains sixty-one horizontal readings separated by 3°.

However, we need more information to completely address our problem. When the robot moves (the global localization problem has been solved, but the re-localization problem starts), we need information about this movement. In order

* Corresponding author. E-mail: fmmonar@ing.uc3m.es

Fig. 1. (Colour online) Manfred V2.

to get it, the robot is also equipped with an optical encoder (HEDS-5540), a sensor necessary to obtain the odometry information. This information gives us an idea about the movement but has several drawbacks: the precision is very low and the error accumulates with the time.

The rest of this paper is organized as follows. First, Section 2 contains a brief review of related work. The localization problem description is presented in Section 3. After that, the evolutive filte is introduced in Section 4. In Section 5, the localization algorithm that we have implemented is explained. The experimental results are presented in Section 6 and, finall , the most important conclusions are summarized in Section 7.

2. Related Work

There are different families of algorithms that can provide a solution to the global localization problem in two dimensions: Bayesian-based methods (grid-based probabilistic filter and the Monte Carlo localization methods can be included here), optimization-based methods (differential evolution (DE) filter and particle swarm optimization filters) and hybrid methods (multi-hypotheses Kalman filters) A large amount of examples can be found in the literature.[2,5–7,9] For example, Burgard *et al.*[2] have studied the problem assuming the robot is in a grid map and Fox *et al.*[6] have developed a Markov localization module in dynamic environments. This paper can be included in the optimization-based field Vahdat *et al.*[28] apply two evolutionary methods (DE and particle swarm optimization) and they compare both of them with the Monte Carlo localization methods.

Most of these methods have been applied to 2D environments. If we talk about the localization problem in a 3D map, we fin less information. For example, Kümmerle *et al.*[11] have worked with the Monte Carlo in outdoors applications. They apply a particle filte to estimate the full 6D state of the robot. These authors solve the localization problem in outdoor environments by matching laser range measurements to a given map of the environment using Multi-Level Surface (MLS) maps.[26] Lingemann *et al.*[13] have developed called High-speed And Yet Accurate Indoor (HAYAI) algorithm that localizes the robot matching features between the data set and the model set. Their contributions lie in fast filterin and extraction of natural features in laser scans and a closed-form solution for computing the pose shift. Tsai *et al.*[27] fuse inertial and ultrasonic sensors and apply an Extended Kalman Filtering (EKF)-based algorithm

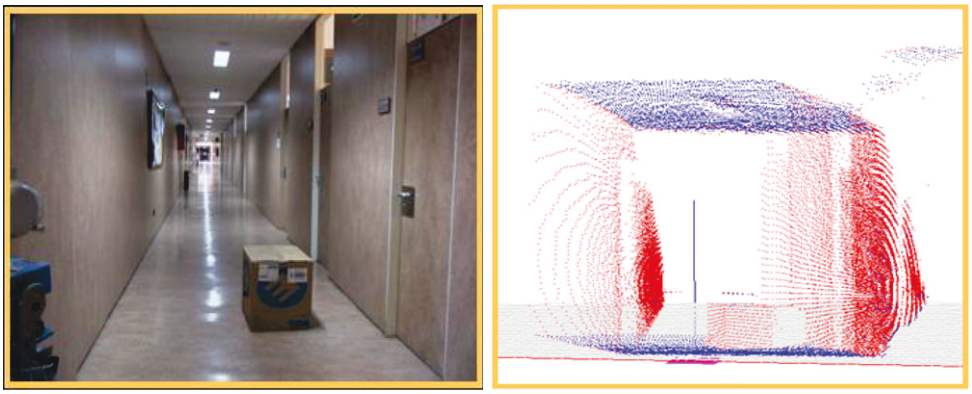

Fig. 2. (Colour online) Corridor of Carlos III University of Madrid and corresponding laser reading.

to estimate the current posture of a mobile robot navigating over indoor uneven terrain. Lai *et al.*[12] have designed a nonlinear method for a system composed of a laser range finde and four artificia reflectors

There are also many groups that are using computer vision techniques.[1,8,21–23,29] Se *et al.*[22] consider global localization as a place recognition problem, and solve it by matching the Scale-Invariant Feature Transform (SIFT)[14] features detected in the current frame to the pre-built SIFT database map. Ho *et al.*[8] demonstrate how the robot can perform global localization using a panoramic mirror in conjunction with a rich 3D model of environment and a particle filte for localization. Royer *et al.*[21] also use a previously built map to compute robot localization in real-time using vision algorithms.

In addition, there are many research groups that are not only focused on solving the localization problem but also concerned with mapping, considering that the robot is in 3D environments. Nuchter *et al.*[20] have developed a robotic mapping method based on locally consistent 3D laser range scans using iterative closest point scan matching. Triebel[26] works with 3D maps building multi-level surface maps. Cole *et al.*[3] use a laser range finde capable of obtaining 3D readings to solve the Simultaneous Localization and Mapping (SLAM) problem. Finally, Thrun *et al.*[25] have built 3D maps of large mines.

3. Localization Problem Formulation and Solution

From the Bayesian point of view, the localization problem can be formulated as a probability density estimation problem where the robot seeks to estimate *a posteriori* distribution over the space of its poses conditioned on the available data. The sensor data can be divided into two groups: $Y_t \equiv \{z_{0:t}, u_{1:t}\}$, where $z_{0:t}$ contains the perception sensor measurements and $u_{1:t}$ contains the odometric information. The recursive determination of the *a posteriori* probability density can be computed in following two steps:

- *Measurement update*. Applying Bayes's rule to the last element of the measurement vector Y_t and assuming that the observation z_t is conditionally independent of the previous measurements given the state x_t, yields

$$p(x_t \mid Y_t) = \frac{p(z_t \mid x_t)p(x_t \mid Y_{t-1})}{p(z_t \mid Y_{t-1})}, \tag{1}$$

  where the denominator of Eq. (1) is obtained by marginalization.
- *Prediction*. The effect of a time-step on the state given the observations up to time t is obtained by observing that

$$p(x_{t+1} \mid Y_t) = \int_{\Re^n} p(x_{t+1} \mid x_t, u_t)p(x_t \mid Y_t)dx_t, \tag{2}$$

  where the assumption that the process x_t is Markovian, and then x_{t+1} is independent of Y_t, has been considered.

Each candidate parameter value in $\Re^n$ yields a value of $p(x \mid y)$, reflectin the posterior probability of the robot pose given the data up to time t. This posterior needs to be weighted according to a given criterion to determine an estimate $\hat{x}$ of the true pose value. Since the *a posteriori* probability distribution is multi-modal in the global localization problem, the minimum mean-square estimate is inconvenient.

The localization algorithm that we have developed, which is probabilistic but not Bayesian-based, concentrates on obtaining the better *maximum a posteriori* (MAP) estimator:

$$\hat{x}^{MAP} = \arg\max_x p(x_t \mid Y_t). \tag{3}$$

This approach is less dependent on statistical assumptions, has a simpler implementation, is robust from a statistical point of view, and has a lower computational cost than the Bayesian methods.

3.1. Localization as a MAP optimization problem

The localization problem is basically an optimization problem, where the robot seeks to estimate the pose that maximizes the *a posteriori* probability density:

$$\begin{aligned}
\hat{x}_t^{MAP} &= \arg\max_x p(x_t \mid Y_t) \\
&= \arg\max_x p(z_t \mid x_t, u_{t-1}, Y_{t-1})p(x_t \mid x_{t-1}, u_{t-1}, Y_{t-1}) \\
&= \arg\max_x p(z_t \mid x_t)p(x_t \mid x_{t-1}, u_{t-1})p(x_{t-1} \mid Y_{t-1}) \\
&= \arg\max_x \prod_{i=1}^{t} p(z_i \mid x_i) \prod_{i=1}^{t} p(x_i \mid x_{i-1}, u_{i-1})p(x_0).
\end{aligned} \tag{4}$$

The MAP estimate expression can be easily stated as an optimization problem subject to constraints (the motion and observation models of the robot).

3.2. Recursive formulation of optimization problem

In order to implement the global localization algorithm in a robot, a recursive formulation is required. The objective function can be reformulated in a more convenient form:

$$\begin{aligned}
f_0(x_t) &= \sum_{i=1}^{t} \log p(z_i \mid x_i) + \sum_{i=1}^{t} p(x_i \mid x_{i-1}, u_{i-1}) + \log p(x_0) \\
&= \log p(z_t \mid x_i) + \sum_{i=1}^{t-1} \log p(z_i \mid x_i) + p(x_t \mid x_{i-1}, u_{i-1}) \\
&\quad + \sum_{i=1}^{t-1} p(x_i \mid x_{i-1}, u_{i-1}) + \log p(x_0) \\
&= \log p_e(z_t \mid x_t) + \log p_v(x_t \mid x_{t-1}, u_{t-1}) + f_0(x_{t-1}),
\end{aligned} \tag{5}$$

where p_e and p_v express the probability density functions for the observation and motion noise, respectively, and the MAP optimization problem can be written as

$$\max_x \log p_e(z_t \mid x_t) + \log p_v(x_t \mid x_{t-1}, u_{t-1}). \tag{6}$$

Then, by perturbing and searching new solutions to Eq. (6), we obtain a recursive version of the MAP estimate.

Algorithm 1 alg_genet_3d

```
1: function alg_genet_3D(...)
2:     for i = 1 : N_P do
3:         estimated_dist_3d(i) ← dist_est_3d(...)
4:         cost(i) ← fitness_3d(estimated_dist_3d(i), real_dist_3d)
5:     end for
6:     while (CONVERGENCE CONDITIONS) do
7:         for i = 1 : N_P do
8:             MUTATION
9:             CROSSOVER
10:            SELECTIONwithTHRESHOLDING
11:            estimated_dist_3d(i) ← dist_est_3d(...)
12:            cost(i) ← fitness_3d(estimated_dist_3d(i), real_dist_3d(i))
13:                    ▷ cost function value calculation for next generation
14:        end for
15:        DISCARDING
16:        [error, ind_best] ← min(cost)
17:        bestmem ← pop(ind_best)
18:        conv_conditions_checking(...)
19:    end while
20: end function          ▷ return bestmem, error and population
```

4. Evolutive Localization Filter

The localization algorithm is based on evolutive optimization techniques. These techniques are probabilistic, but without derivatives or probability density functions to estimate the best solution to the localization problem. In our evolutive algorithm, there are elements that correspond to possible solutions, and the fitnes function value represents the error between the real and estimated data. The stochastic search of the robot's coordinates is done using the DE method proposed by Storn and Price[24] for global optimization problems over continuous spaces, which is explained in this section.

In order to do that, the reader can see Algorithm 1. Besides, a complete explanation of the DE algorithm can be found in our previous work.[19]

First of all it is necessary to comment on some details about the environment and the robot's characteristics.

The environment has been modeled geometrically as an occupancy grid map in three dimensions and the robot's pose (the robot's pose is define as the robot's position and orientation: x, y, z, and yaw) is represented with the cartesian coordinates and the horizontal orientation. In most of our experiments, each cell is a cube of 12.1 cm side and the whole map contains $500 \times 119 \times 25 = 14, 87, 500$ cells. It represents a 3D environment with a 871 m^2 area and a height of 3 m. If we want to determine the robot's localization, four coordinates must be estimated $(\hat{x}, \hat{y}, \hat{z}, \hat{\theta})$, definin a state space with 4 d.o.f in a 3D map.

The simulated environment and the whole algorithm have been implemented using MATLAB, which is a numerical computing environment and 4th generation programing language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, and Fortran.

The search starts with a population of N_P candidates (the algorithm is population-based and N_P represents the number of elements), which are introduced in the localization module and evolve with time to the best solution. Each candidate is a possible solution to the global localization problem (the robot's pose, with 4 DOF). The initial population will be chosen randomly.

For each candidate, its associated fitnes function is calculated (line 2 to 5 in Algorithm 1). The fitnes function is a key component of the method. It compares real data received in a laser reading with simulated data from a candidate solution. The estimated data have been sorted to configur the observation vector, with 13 vertical scans separated by 5° and 61 horizontal readings separated by 3°.

The main loop starts in line 6. If one of the convergence conditions is satisfied the localization process ends successfully.

Another loop, which contains the evolutive search, starts in line 7. It consists of a generation of a new population for the next generation. In a single iteration the algorithm is executed to obtain the next candidates, evolving with time to the correct pose. Each part of the mechanism is explained in the next subsection.

4.1. Mutation, crossover, and selection

The initial population is perturbed to generate a variation v according to the following expression:

$$v = x_a^k + F\left(x_b^k - x_c^k\right), \tag{7}$$

where x_a^k, x_b^k, and x_c^k are parameter vectors chosen randomly from the population at iteration k and are different from running index (this x is not the cartesian coordinate but the element of population, define by four coordinates). F is a real and constant factor that controls the amplificatio of differential variations $(x_b^k - x_c^k)$. An example of the perturbation process can be seen in Fig. 3. There is an initial parameter vector x_i^k and the perturbation is done with three

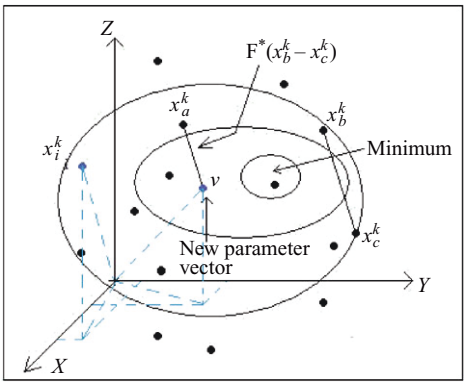

Fig. 3. (Colour online) New population member generation.

random variables and the constant factor F, generating the new parameter vector v.

The variables of this expression could be changed in order to obtain a different perturbed vector. It is possible to select the *best* population vector instead of x_a^k. The current implementation allows both possibilities: random mutation and mutation from the best candidate. The firs choice is more robust because it maintains the population diversity and is applied in general situations. However, the mutations from the best candidates are faster and they could be applied in some situations, such as the robot is situated in a place that is very easy to identify, or tracking tasks after a successful localization.

In order to increase the diversity of the new generation, the crossover is introduced. Denoted by $u_i^k = (u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,D}^k)^T$, the new parameter vector is

$$u_{i,j}^k = \begin{cases} v_{i,j}^k & \text{if } p_{i,j}^k < \delta \\ x_{i,j}^k & \text{otherwise} \end{cases}, \tag{8}$$

where $p_{i,j}^k$ is a randomly chosen value from the interval [0, 1] for each parameter j of the population member i at step k, and δ is the crossover probability and constitutes the crossover control variable. The random values $p_{i,j}^k$ are made anew for each trial vector i.

The new member of the population u_i^k is compared to x_i^k to decide whether or not vector u_i^k should become a member of generation $i + 1$. If the vector u_i^k yields a better value for the objective fitnes function than x_i^k, then it is replaced by u_i^{k+1}; otherwise, the old value x_i^k is retained for the new generation. The general ideas of the previous mechanism (mutation, crossover, and selection) are well known and can be found in literature.[7]

4.2. Thresholding mechanism

Evolutionary Algorithms (EAs) and, in general, population-based methods, have become very popular due to its applicability and implementation simplicity. One of the most important shortcomings of these methods is the premature convergence and the lack of robustness in noisy optimization problems. If DE is compared with other population search-based methods (for example, Genetic Algorithms (GA)), it shows some weakness. This behavior has been studied by Krink *et al.*[10] There are two different aspects with a negative influence the DE method implements a greedy search strategy and the DE mechanisms for generating new potential solutions are less stochastic than other EAs.

These disadvantages are significan when the difference between the candidate solution fitnes value and the current population element fitnes value is smaller than the fitnes variance originated by the noise.

The idea of thresholding is to reduce the eagerness of the algorithm by rejecting those new solutions that do not improve the previous hypothesis in a pre-specifie magnitude τ. This idea is not new and has already been applied to evolutionary computation problems.[15] The threshold can not be a fi ed magnitude, because this unit depends on the noise variance and the fitnes distance to the optimal fitnes value. The noise variance might be estimated, but it is not easy to estimate the second factor.

First, the fitnes value difference between the population member x_i^k and the candidate member u_i^k is calculated. Then, this difference is compared with a predefine threshold value τ in order to determine if the improvement shown by u_i^k is not caused by the noise. If this condition is met, the target vector x_i^k is replaced by u_i^k in the next generation; otherwise, x_i^k is kept into the population.

In conclusion, if the improvement in the fitnes function is bigger than the variance (or the standard deviation, depending on the selected units), it can be considered that it is not caused by the noise, and the new member can be introduced in the population.

The algorithm convergence and robustness in noisy optimization problems and the comparison between the current method and the initial method without thresholding have been published in our recent work.[16]

In our experiments, we have chosen a value that depends on the sensor noise because the thresholding mechanism tries to avoid optimization in the noise band. The threshold τ is equal to $S_N \times f_i^k$, where S_N is the sensor noise (percentage over the distance weighted) and f_i^k is the fitnes function value for the ith population member in the kth iteration. This value has been chosen empirically. For example, when the sensor noise is 3% over the distance weighted, a simple and adaptive threshold level of $\tau = 0.03 \times f_i^k$ has been adopted to reject the offspring solution generated by f_i^k. In spite of its simplicity, it works quite efficientl . This selection mechanism decreases considerably the eagerness of the DE algorithm and also its speed of convergence. As a consequence of thresholding, the algorithm rejects a high quantity of new solutions and accepts only those solutions that present a clear improvement in the fitnes function. The stochastic robustness of the algorithm, its computational cost, and the iterations to converge present worse values, so another mechanism (discarding), must be incorporated to reduce this negative influence

4.3. Discarding mechanism

The use of a thresholding band tends to decrease the convergence speed of the algorithm, particularly at initial stages due to the rejection of offspring that does not improve enough the previous hypothesis (below the threshold band).

A discarding mechanism has been introduced to increase the speed of the algorithm while maintaining the stochastic advantages in terms of robustness of thresholding. The idea is to determine the worst fitnes individual of the new population and substitute it by a new solution close to a better one. In order to do that, a percentage of elements to be discarded is chosen, not only one (for example, 5%). In order to avoid concentrating the discarded solutions around the best existing individual, one of the members of the population with its fitnes value located in the firs half of the fitnes ranking is selected randomly. This selected solution plus a relatively small random component is adopted as a new offspring.

The discarding mechanism is important at early stages of the optimization process, where errors are important, and it is not so interesting at fina steps, where the population is close to the solution.

4.4. Convergence conditions

There are many possibilities when choosing a suitable objective function to solve the recursive optimization problem given by Eq. (6). Assuming that the sensor measurements are Gaussian-distributed, one of the most common choices as a basis for the estimate is the L2-norm. In this case, the objective function to be minimized by the evolutive algorithm for a point of the state space included in the population x_t^j could be given by the following expression:

$$f_0^j\left(x_t^j\right) = \sum_{i=0}^{N_s} \frac{\left(z_{t,i} - \hat{z}_{t,i}^j\right)^2}{2\sigma_e^2} + \frac{1}{2}\left(x_t^j - \hat{x}_t\right) P^{-1} \left(x_t^j - \hat{x}_t\right)^T, \tag{9}$$

where $z_{t,i}$ is a single measurement contained in a reading given by the 3D laser at instant t, $\hat{z}_{t,i}^j$ is the expected observation for that measurement if the robot was situated in x_t^j, $\hat{x}_t$ is the estimated pose (if it exists, in time t), P^{-1} is the covariance matrix of the state error, and σ_e^2 is the observation error variance. We have considered that the laser error is Gaussian-distributed over the weighted distance. The second term of the expression depends on the estimated pose $\hat{x}_t$, which does not exist at the beginning, and could not be unique, because the robot could start from different poses. It is straightforward that the second term of the right part could be estimated as a function of the distance between the candidate pose and all the viable ones, but it will complicate too much the fitnes function evaluation, adding also a statistical component.

In order to accelerate the calculation, we will not include in the fitnes function the information given by the distance between the candidate pose and the estimated pose until the algorithm has converged to one single pose (it is considered that the algorithm has converged when all candidate poses are in a sphere of constant ratio around the best one). Excluding this term, the fitnes function before convergence will be given by the following expression:

$$f_0^j\left(x_t^j\right) = \sum_{i=0}^{N_s} \frac{\left(z_{t,i} - \hat{z}_{t,i}^j\right)^2}{2\sigma_e^2} = \frac{1}{2} \sum_{i=0}^{N_s} \frac{v_{t,i}^2}{\sigma_e^2}, \tag{10}$$

where $v_{t,i} = \left(z_{t,i} - \hat{z}_{t,i}^j\right)$ represents the discrepancy between the observed and predicted sensorial data.

It would be interesting for us to fin an expected value for the cost function. In order to do this, we have noticed that there are two effects that influenc the cost function: the measurement noise and the estimation error. If we obtain a perfect estimation, the second effect can be eliminated, but an error introduced by the noise will always exist. Hence, it is possible to estimate the expected value of the objective function when it is close to the true value $E(f_0)$. We have to remark that it is the fitnes function value in the real position.

If we observe the term $\sum_{i=0}^{N_s} v_{t,i}^2/\sigma_e^2$ of the last expression, the components $v_{t,i}^2/\sigma_e^2$ are random variables with a standard Normal distribution $N(0, 1)$, and the sum follows a Chi-Square probability distribution with N_s DOF. This probability distribution, which is well known and tabulated, has an average of N_s and a variance of $N_s/2$. Therefore, the expected value of the objective function that we are trying to minimize is

$$E[f_1] = \int_{-\infty}^{+\infty} f_0(v) p(v)\, dv = N_s/2. \tag{11}$$

This expression tells us that even if the pose that we are evaluating was the correct pose of the robot, and due to measurement errors that occur in the sensor during the environment perception, the expected value of the objective function is $N_s/2$.

The determination of a stopping condition for the algorithm brings us back to what was discussed above, because once we know the best expected value of the objective function, it is possible to establish a stopping condition based on some of the statistical parameters associated with this objective function.

As we have concluded that our cost function can be approximated by a Chi-Square with N_s DOF, it is straightforward to associate the objective function value with a given probability. In other words, an objective function value f_{1-p} with probability $1 - p$ means that the optimum value has a probability $1 - p$ of being below f_{1-p}. These values are often found tabulated in statistical literature for some typical quantiles and a given number of DOF.

Besides, we have chosen other simple criteria that have been used experimentally in certain situations. This criteria do not ensure convergence but may lead to good results in less time. If one of the following convergence conditions is satisfied the localization process finishes

- Number of iterations without changes in the fitnes function value of the best estimation is bigger than a constant.
- Number of iterations without changes in the fitnes function value of the worst estimation is bigger than a constant.
- Number of iterations without changes in the difference between the fitnes function value of the best estimation and the fitnes function value of the worst estimation is bigger than a constant.

4.5. Fitness function

The cost function compares laser data obtained from the robot true pose to laser data from the pose estimates. The localization algorithm minimizes the error and evolves to the true solution. The sensor considered in this paper is a simulated laser that works in 13 vertical scans separated by 5°, and each vertical scan contains 61 horizontal readings separated by 3°. The localization method applies the fitnes function aligning the current scan (estimated pose) with the reference scan (real pose). The population evolves to the correct location because the matching error is minimized.

The matching method applied in this paper avoids searching for point associations by simply matching points with the same bearing. It is called Polar Scan Matching (PSM)[4] and the different options implemented are explained below.

The natural choice for the cost function is the sum of the squared errors function (L2-norm). As explained in the previous section, if the observation vector at time t is $z_t = (z_{1,t}, \ldots, z_{p,t})^T$, the predicted observations according to the estimated robot's pose are $\hat{z}_t = (\hat{z}_{1,t}, \ldots, \hat{z}_{p,t})^T$, and the observation error variance is σ_e^2. Then the penalty function for a single point x_t^j of the population can be stated as

$$L2\left(x_t^j - \hat{x}_t\right) = (z_t - \hat{z}_t)^T (z_t - \hat{z}_t) = \sum_{i=0}^{N_s} \frac{\left(z_{t,i} - \hat{z}_{t,i}^j\right)^2}{2\sigma_e^2}. \tag{12}$$

Following are some of the factors in global localization that make this fitnes function difficul to manage:

- The range and accuracy of the sensor and the number of sensors limit the possibility of distinguishing between different poses, leading the fitnes function to a high number of global maxima.
- The geometrical similarities in the environment due to the repetition of the space distribution originates the presence of a high number of possible robot's pose solutions to the mean square loss function.

The loss function define in this way provides us with an optimization mechanism that obtains an unstable parameter solution and, by extension, an unstable filte . The evolutive filte can move from one local minimum to another, originating abrupt changes in the pose estimate. This problem comes from the fact that we are not using all the information we know about the system in the loss function. In fact, we only use the observation model to predict the sensor measurements at each position, and these predicted measurements together with the actual measurements are introduced in the loss function to evaluate the robot's pose estimate. The instability originated by the existence of multiple solutions to the loss function cannot be solved by considering only the available sensor observations.

Besides, the least squares method is not completely satisfactory when the noise model is not exactly known or the model is contaminated with other probability distributions.

The absolute error (L1-norm) may be an appropriate measure in some situations. For example, it presents a better performance with high errors originated in outliers or contaminated measures. If the L1-norm is not derivable, then it is necessary to apply linear programing methods to obtain a solution to the optimization problem. Based on the Monte Carlo studies, the use of the L1-norm has been recommended when the errors follow one of these distributions: Laplace, Cauchy, mixture of normal and uniform, and contaminated normal.

If the fitnes function selected is the L1-norm, the function to minimize is given by the next equation:

$$L1\left(x_t^j - \hat{x}_t\right) = |z_t - \hat{z}_t| = \sum_{i=0}^{N_s} \frac{\left|z_{t,i} - \hat{z}_{t,i}^j\right|}{\sigma_e} = \sum_{i=0}^{N_s} \frac{|v_{t,i}|}{\sigma_e}, \tag{13}$$

where $v_{t,i} = (z_{t,i} - \hat{z}_{t,i}^j)$ represents the discrepancy between the observed and the predicted values of the sensor data.

5. The Localization Method

We have developed an improvement of the ELF algorithm that solves the global localization problem in a robust and efficien way using 3D sensor data. A simulation of the localization method implemented in the robot can be seen in Algorithm 2.

The robot is located at a place in the environment. This location is represented by three spatial coordinates and the orientation in the horizontal plane. It is important to remark that our method works in three dimensions, but there are cases where we can assume the hypothesis that the localization could be solved in two dimensions. For instance, when the environment is plane and the robot has a fi ed height, we can consider the z coordinate as a constant value.

We do not have any information about the location of the robot, thus the place can be considered randomly. The mobile robot receives the information in a 3D laser reading. Our method is now working with simulated data, and the reading is done in line 2 of the algorithm from the real location of the robot.

After that an initial population is generated, covering the map randomly (line 3). That population contains N_P possible candidates.

When the laser reading has been done, the main loop starts (line 5), working in the following way: the robot tries to locate itself on the map (we have called it *step*) and, in order to do that, the evolutionary algorithm (line 6) is executed, returning the estimated solution, the error, and the fina population.

The concept *step* will be used several times, and we will explain what it means in our context. A *step* can be regarded as the moment in time at which the robot uses a single 3D laser scan to locate. When it receives another scan and motion information, a new *step* starts. In other words, we say the robot is located in a single *step* when the it uses information from a single 3D laser scan, without considering movement information. If the robot moves to another site and takes another scan, it uses more information to locate (the second laser scan and the movement information). In that case we say it is in the second *step*.

Algorithm 2 Localization_3D

```
1: PARAMETERS INTRODUCTION
2: real_dist_3d ← dist_est_3d(. . . . . .)          ▷ Laser simulation
3: pop ← start_pop(. . . . . .)
4: dir_desp ← NULL
5: while dir_desp ≠ end do
6:     [bestmem, error, pop] ← alg_genet_3d(. . . . . .)
7:     move_robot ← VALUE                  ▷ robot's displacement
8:     bestmem ← bestmem + move_robot
9:     real_position ← real_position + move_robot ∗ (1 + error_pos)
10:    real_dist_3d ← dist_est_3d(. . . . . .)          ▷ Laser simul.
11:    for i = 1 : N_P do
12:        pop(i) ← pop(i) + move_robot ∗ (1 + error_pos ∗ random(1))
13:    end for
14: end while
```

Once the localization process in a *step* has finished the robot starts moving. The robot location and the best estimation are moved according to that displacement (with an added error, lines 8 and 9) to integrate the movement information. The sensor makes a new laser reading (line 10) and receives odometry data. Then the possible locations are displaced according to the odometry (with an error, too, lines 11–13) and the localization process in the second *step* starts, but now with the results of the firs one in the initial population. It is obvious that the localization process is easier in this case because there is more information. When the second *step* finishes the third *step* starts, and so on.

The algorithm is running continuously while the robot moves within the environment. We have simulated that the robot has completed its task introducing an *end* in line 5. According to Algorithm 2, the loop finishe when an *end* is introduced in the movement variable, which means the robot is turned off or the movement is over and the robot is successfully localized.

It is necessary to defin several parameters before the execution. The most important ones, with a critical influenc on the behavior of the algorithm, are listed below:

- Population Size
  It is also necessary to introduce the initial number of elements. It is critical and depends on the environment size. When the robot is localized, the needed number is smaller, and we can reduce it with a positive influenc in the computational cost.
  Another problem is how to determine the population number for a given environment. On the one hand, the number depends on the map size. On the other hand, there is another important element: the symmetries. The symmetries originated by the robot's position and its perception capabilities can multiply the potential number of equivalent maxima. This number can vary from one in the most favorable case to more than 50, as we have observed in our experiments. The required population number can change dramatically because a certain number of population elements are required to maintain an hypothesis until new data are perceived.
  It is commonly known that nonlinear population-based global optimization algorithms can easily and prematurely converge to point estimates that are not globally optimal. It can be originated by a lack of population diversity, the slowness of the search algorithm, or the existence of a local minima in a multi-modal objective function. The firs two problems can be addressed by increasing the population size, but the third one is more difficult First, the function can be multi-modal, so the algorithm will not converge until the multi-modality disappears. Second, if the noise level rises, a fictitiou multi-modality induced by the noise level can appear.
- Mutation Amplificatio Factor (F) and Crossover (δ)
  F is a real and constant factor that controls the amplificatio of differential variations in the perturbation of the parameter vector (mutation). In this paper, the experiments are done with $F = 0.85$. The value of δ controls the crossover and is fi ed to 0.75 experimentally. Both variables were explained in Subsection 4.1.
  We have also adopted an adaptive adjustment of the perturbation amplificatio factor F. This mechanism tries to maintain a high amplificatio factor, while the population has not converged to the promising areas (a wide scope search is required) and to limit the algorithm search scope when the population set is distributed in the most feasible areas.
- Upper Limit of Iterations
  There is also an upper limit for the number of iterations per *step*, which is an added feature designed to track the robot faster when it is not necessary to localize it in a single one. It is faster because we use more information (several laser readings and odometry information), and it can be changed to obtain different results. For example, 15 iterations per *step* make the algorithm go faster, but we need more *steps* to obtain a good localization. Nevertheless, 500 iterations make the algorithm go slowly, but we can localize the robot in a single *step*.
- Tracking
  Once the localization process has finishe in a satisfactory way, the robot knows exactly its position. Our problem is now converted into the tracking problem. In other words, the robot will need to be continuously localized, updating its pose after small changes of position.
  In order to do that online, the population number is reduced drastically when the fitnes function of the best

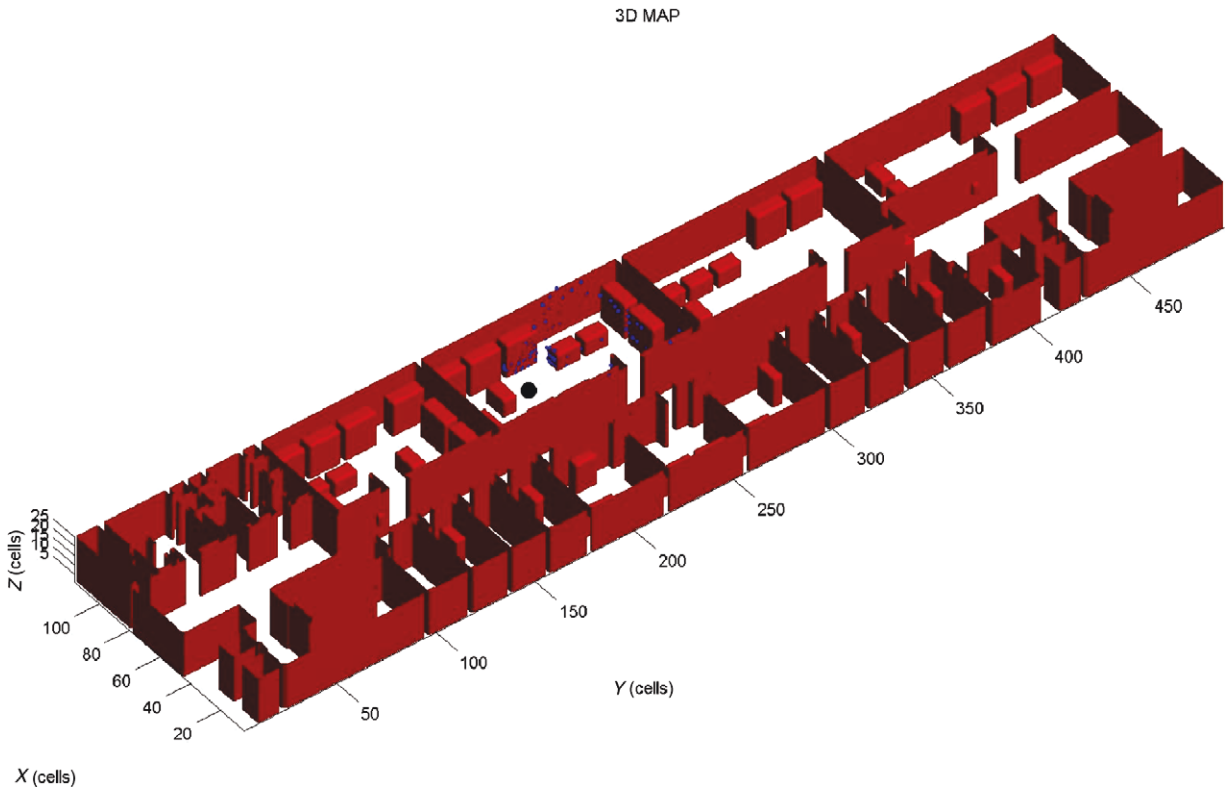

Fig. 4. (Colour online) Three-dimensional map.

estimation is under a threshold, which means that the robot is localized. We have chosen 10 as the population size after that. This reduction implies an impressive advance in the computational cost. In each motion cycle the robot can now update its pose accurately and quickly.

6. Experimental Results

All experiments have been developed in a simulated indoor environment: laboratories, corridors, and office of the Carlos III University of Madrid. An example of a 3D map can be found in Fig. 4. The measurement model is the same that was explained in Section 4.5.

6.1. Effectiveness of the algorithm

In the firs experiments, we have tested the capacity of our method to solve the localization problem. In order to do that, we must tackle the problem of localizing the robot in a random place of the environment only with the information of sensors at this point (3D data obtained with the laser range finde and movement information).

First, we will consider that the robot is still. The way to check it consists on running the algorithm a number of times, situating the robot at different places, and obtaining the percentage of success. The global localization process starts with the robot situated in a random place of the environment and two different cases and variables defin the process: the estimate matches the real pose (the variable *true* is incremented by one) or the estimate and the real pose do not coincide (the variable *false* is incremented by one). After that, a new random place is generated and the previous process (which is define as a single trial) is repeated. The total number of trials is *total*. The percentage of success is equal to *true* divided by *total*. We have chosen 100 as the number of elements of the population. The laser range finde has an error of 3% over the distance weighted. We have not considered the convergence of the algorithm in this experiment, and we let the algorithm run a high number of iterations (500) to see whether it can obtain the solution or not.

The success of running the program under these conditions is 84%. This percentage depends on the environment and our map contains many similar places whose correspondent measurements are the same. Therefore, it is not possible to distinguish between several different locations because the sensor measurements are the same. For example, it occurs when the robot is inside an office That is why the algorithm does not succeed in some cases and the localization method converges to all possible solutions (correct place and similar locations). We consider it as a failure because it does not converge to a single solution. The algorithm returns one solution but it could be a wrong place. It can also be explained using the human beings perception. If you are in an offic corner looking at the wall it is not possible to know in which offic you are. It can be concluded that the information is not good enough to apply localization techniques in these cases.

This is a promising result, because we have not included the movement of the robot (using single *step* to localize the robot, with information of one 3D laser scan and no motion information), but it does not reflec a real situation. The purpose of the experiment is to check the capability of our method to solve the global localization problem. Therefore, the RELF-3D algorithm is suitable for solving this. An example of the information that the robot would receive is shown in Fig. 5, where the 3D environment, the laser readings, and the robot are represented.

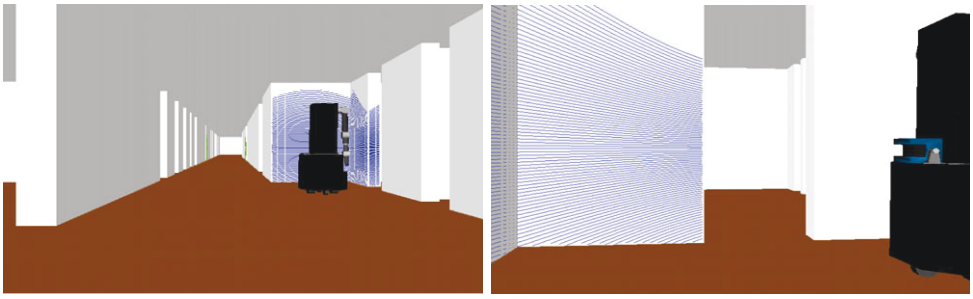

Fig. 5. (Colour online) Example of localization in a corridor.

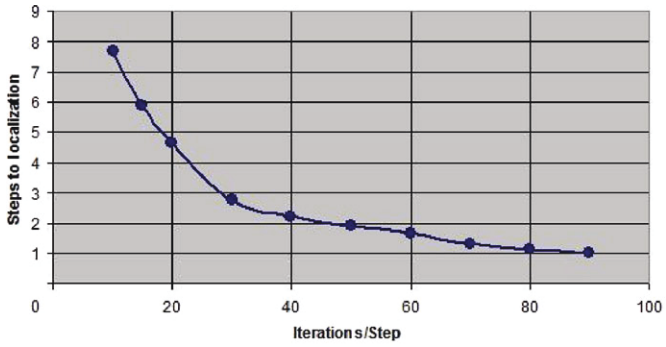

Fig. 6. (Colour online) Steps to localization vs. iterations/step. Error = 3%, convergence criteria: worst member fitnes value smaller than $3 \times N_s/2$. Algorithm options: random mutation with thresholding and discarding. $N_s = 100$.

However, we will not fin this situation in real life, because the robot will not be still. We have also tested the capacity of our method under movement, making different experiments with changes in the maximum iterations per *step*. On one hand, with a high number of iterations per *step*, we will be closer to the solution in one single *step*, but with a lower velocity because we need more time to deal with a single laser reading. As a reminder, the robot uses the firs laser reading to locate, with a given number of iterations, and then the second laser reading from another location is introduced in the localization module, including motion information. On the other hand, with a low number of iterations, the robot will need less time to execute a *step*. The number of *steps* to localization against the iterations per step is shown in Fig. 6.

The laser range finde has an error of 3% over the distance weighted and the population size is 100. The convergence criteria to conclude that the robot is successfully localized is that the worst member fitnes value is lower than $3 \times {}^*N_s/2$. The average number of *steps* after situating the robot at different places is shown in the figure

Interesting results can be concluded observing Fig. 6. It needs more *steps* to succeed with a maximum of *10* iterations per step (lower limit chosen). Therefore, we obtain better results in other cases. The time needed (it depends on the number of iterations) to execute the localization algorithm in a *step* is less, but we do not know the situation of the robot until eight *steps*. It means that the number of laser readings used, and also the distance from the initial location to the current one, is bigger. This could be quite dangerous for the robot, because it goes a long way before being properly localized on the map. If the number of iterations per *step* is 90, the robot will need one single *step*. However, the time needed could be a problem if the time per *step* is a constraint. It can be concluded that different values can be chosen depending on robot requirements. For example, good results are obtained with 40 iterations per *step*. The robot is successfully localized quickly in two steps using a low number of iterations.

Vahdat *et al.*[28] show a similar experiment implementing three different methods: DE, particle swarm optimization, and Monte Carlo. However, the number of iterations per *step* is surprisingly low. They need two or three iterations in each *time-step* and about 25 *steps* for robust convergence. The experiment is different from ours because it does not limit the number of *steps* but the number of iterations. It can be because it is not dangerous for their robot and it can wait 25 *steps* until convergence. It is a different result that complements our experiments. They compare convergence rates, convergence times, and individuals needed for convergence, and their conclusion is that DE converges faster, more robustly, and with fewer individuals.

The percentage of success running the program under these conditions is higher (93%). This is a logic result, because we have more information: several laser readings and movement (odometry) information. It means that the algorithm returns to the correct location in the majority of cases, a very promising result considering that the places are random and there are many symmetries in the map.

6.2. *Accuracy of the method*

The localization process results in different points of the environment can be seen in Table II. The cartesian coordinate of the point we want to localize in the map is x, and $\hat{x}$ is the estimation obtained. This can be extended to y, $\hat{y}$, z, and $\hat{z}$. The orientation in the horizontal plane (tilt) is θ and $\hat{\theta}$ is the estimation obtained. The sensor noise level is 3% (a Gaussian distribution over the laser distance with a standard deviation of 3%). The position error (e_p) is the distance in millimeters between the estimated value and the real position of the robot. The orientation error (e_θ) is the difference between θ and $\hat{\theta}$, in degrees.

We can conclude that the accuracy of the localization process is improved significantl compared with our previous works in 2D maps, and we believe it is the most important advance that we have obtained. If we observe the position error column (e_p), the average error is equal to 5.1 mm, and the same measurement with our previous algorithm using 2D sensor data was equal to 2 cm, as can be checked in ref.

Table I. Errors localizing the robot at different points. The cartesian coordinates are in cells and the errors in millimeters and degrees.

	x	y	z	θ	$\hat{x}$	$\hat{y}$	$\hat{z}$	$\hat{\theta}$	e_p (mm)	e_θ (deg)
1	47	358	4	0	47.001	358.000	4.039	0.000	4.76	0.000
2	75	215	14	90	75.004	214.984	13.991	90.016	2.24	0.016
3	13	260	13	10	13.000	260.005	12.991	9.997	1.21	0.003
4	82	240	16	45	82.016	240.004	16.047	44.957	6.06	0.043
5	40	385	15	5	40.015	384.997	14.976	4.981	3.50	0.019
6	48	68	10	180	48.006	68.027	10.009	180.017	3.51	0.017
7	60	60	7	5	59.993	60.015	7.032	5.032	4.31	0.032
8	82	100	11	90	82.015	100.000	11.000	89.985	1.88	0.015
9	60	380	6	0	60.001	380.005	5.972	0.000	3.41	0.000
10	110	130	9	270	110.009	130.007	9.018	269.937	2.64	0.063
11	70	23	9	0	70.001	23.045	9.066	360.000	9.63	0.000
12	110	260	3	250	109.990	260.021	2.915	250.016	10.63	0.016

[19]. The main reason for this reduction is the increase in the amount of information. The 3D laser reading contains nine bi-dimensional scans. Thus, the information is multiplied by nine.

It is possible to compare the accuracy results with those obtained by other groups working on localization in 3D environments. The accuracy achieved by Kümmerle *et al.*[11] is about a few centimeters. Se *et al.*'s[22] average Euclidean translation error is equal to 7 cm. and the average rotation error is 1º. However, they mention that it can be reduced by using a higher image resolution. Ho *et al.*[8] localize the robot within 1 m. of true position and 5º for orientation. Their best results are 9 cm and 0.49º. The localization error mean obtained by Royer *et al.*[21] is equal to 1.9 cm and 0.1º. The results obtained in this paper are more accurate than those approaches.

Furthermore, this improvement has an important consequence for the computational cost. We work with a 3D laser scanner and it has a negative influenc on the computational cost. However, the accuracy is higher and we need less iterations to obtain satisfactory results. The negative influenc of the 3D laser data can be partly compensated with the positive influenc of the accuracy improvement (allowing us to use the algorithm faster with relaxation in the accuracy or convergence requirements).

If we observe the orientation error, we see that its value is very small. The orientation error average was equal to 0.02° for the results shown in Table I. Therefore, it can be concluded that the method presents good features in this aspect.

The influenc of the sensor noise level can be observed in Fig. 7, where the localization error (in centimeter) is represented versus the laser noise level (standard deviation of a Gaussian distribution over the laser distance, in percentage of the weighted distance). We obtain acceptable results even with a high noise. It is a great result that leads us to conclude that the algorithm presents good performance in this aspect.

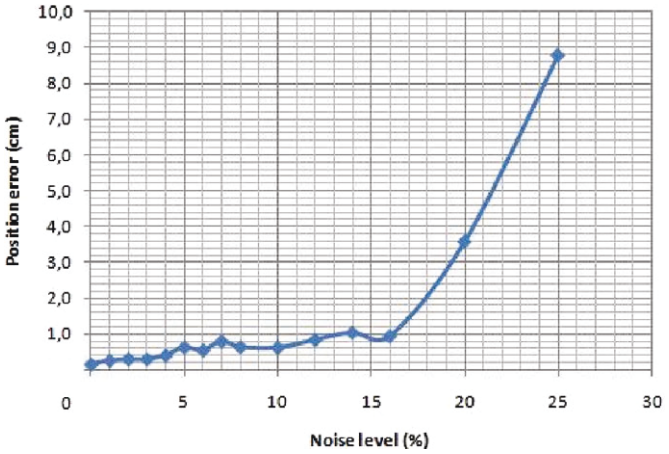

Fig. 7. (Colour online) Error vs. sensor noise level.

6.3. *Thresholding and discarding effect*

A threshold rejection band has been added to avoid the premature elimination of solutions. This mechanism decreases the eagerness of the algorithm, letting it to eliminate a candidate solution from the set only when the offspring candidate is significantl better from a statistical point of view. A discarding operator was also added to accelerate the convergence of the algorithm by eliminating a very low percentage of the worst population individuals at each iteration of the algorithm, and re-sampling the individual candidate in the vicinity of a better candidate selected randomly between the best elements of the population.

An important effect of the use of a threshold rejection level is the decreasing of the convergence speed, as shown in Fig. 8. The latest improvement occurs in iteration number 380 in the thresholding option for the best estimation, while it is produced around iteration number 260 in the other case without thresholding. The fitnes function is the L2-norm (Eq. 12), and the observation measurements are in cells.

The use of a discarding mechanism accelerates the convergence of the algorithm with a threshold level. Studying the case with thresholding and discarding, it can be concluded that the convergence is reached earlier.

The algorithm with discarding is even faster than the basic version, while maintaining the capability to avoid premature convergence inside the noise band, as can be seen in Fig. 9, which shows the solution of the algorithm in a highly symmetrical environment.

In this type of environment it is not possible to distinguish between all rooms, because the observations are the same for all of them. If the algorithm is optimized in the noise band,

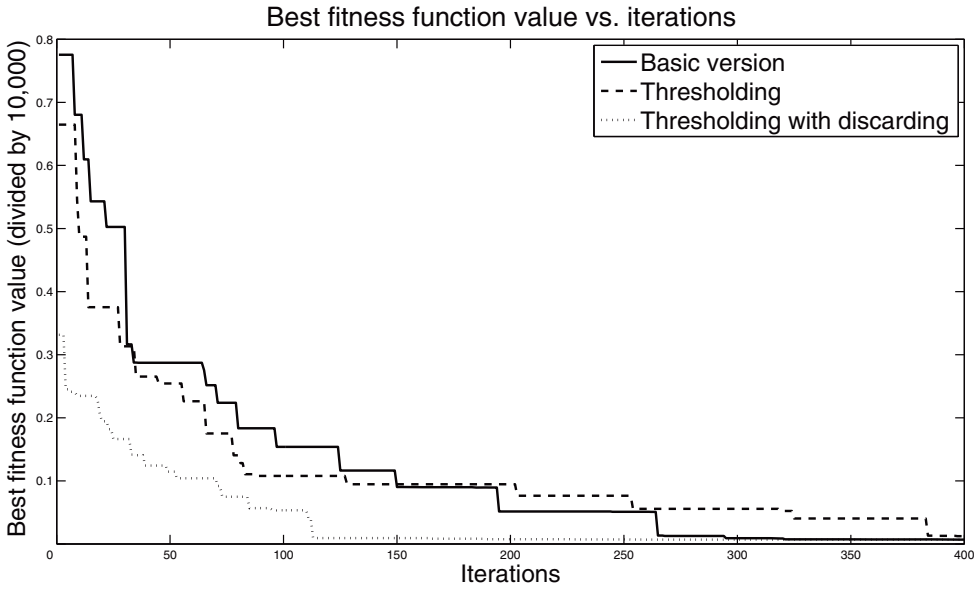

Fig. 8. Best fitnes evolution value vs. iterations. Three different options – without thresholding, with thresholding, and with a tresholding band – with 5% of population discarding.

it will eventually converge into a room, giving us a wrong result. It will finall converge to one solution, eliminating all other rooms and concentrating all the population around the optimal solution.

It can be observed that the estimated solution (point in the center, with coordinates (30.00, 385.00, 5.97, 0.00)) does not coincide with the real solution (30, 220, 6, 0), but there are still candidates in all rooms. It is a logical result that indicates that the algorithm avoids the premature elimination of solutions.

Is is also important to say that the thresholding mechanism does not always make the algorithm converge slower. There are cases where the noise makes the algorithm evolve toward erroneous solutions. Although the number of candidates that evolve is smaller, the improvements are more significant and the big influenc of the best candidates on the population improves the convergence speed. It occurs in situations where noise plays a very important role, as can be seen in Fig. 10, which shows the evolution of the best fitnes function value in a particular case under the influenc of an important noise (we

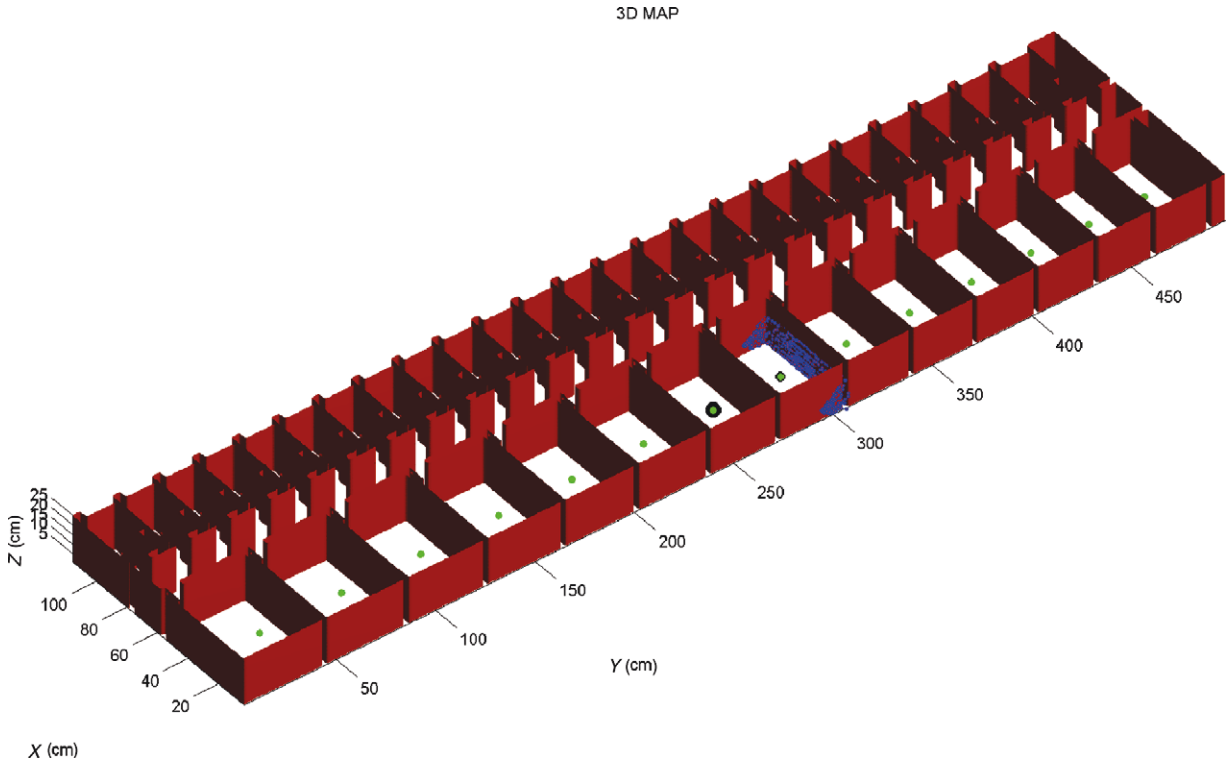

Fig. 9. (Colour online) Algorithm solution in a symmetrical environment with thresholding and discarding.

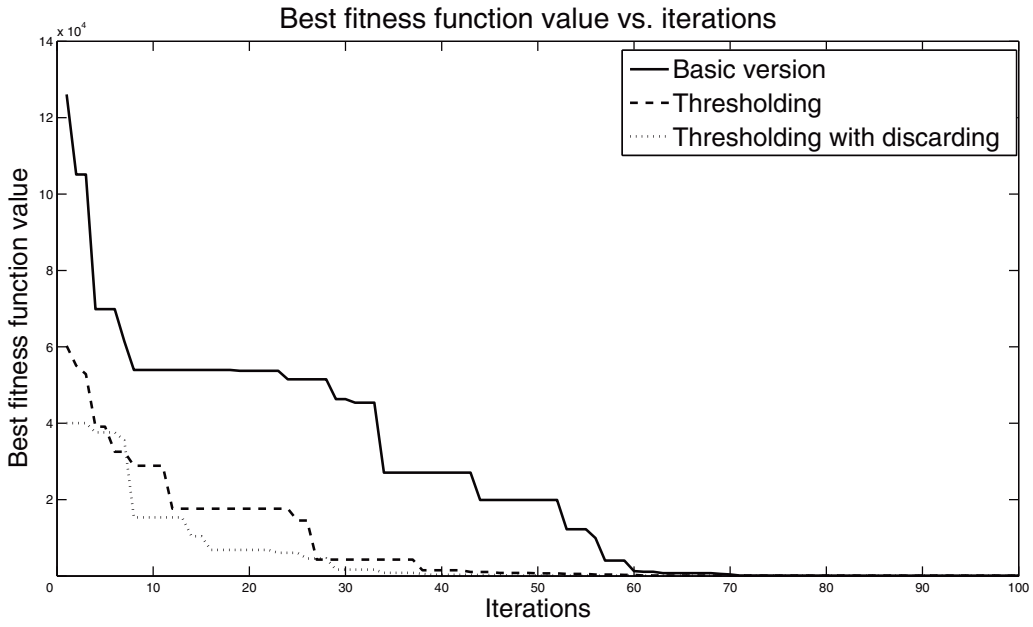

Fig. 10. Best fitnes function value vs. iterations. Three different options: without thresholding, with thresholding, and with a tresholding band with 5% of population discarding. Noisy case (10% over distance).

use a range laser finde as sensor, and the noise introduced is equal to 10% over the distance weighted). It can be observed that thresholding increases the convergence speed in this case, being even close to thresholding with discarding option.

6.4. *Fitness function options*

There are several options when choosing an appropriate cost function for our particular problem. The L1-norm and the L2-norm have been studied in this paper, and the results are presented below.

In order to test the algorithm characteristics in this aspect, a simulated environment has been considered (Fig. 4). This environment is similar to many offic indoor areas where all office are located along the central corridor. This test environment will be used to verify the accuracy and robustness properties of the evolutive localization filte algorithm under two different cost functions, and the effect on the algorithm convergence of a contamination in the noise error. A Gaussian observation error of zero mean and standard deviation σ equal to 3% of the measured distance has been considered.

The firs test tries to determine the capability of the algorithm to localize the robot when it is located in one of the distinguishable rooms. The robot's pose is variable and there is no contaminated noise. The position errors in centimeters are presented in Table II (columns with subindex NC).

The second test tries to determine the capability of the algorithm to localize the robot when there exists a contaminated Gaussian noise. This behavior has already been studied in our previous work.[18] This situation happens when there are mobile objets or unexpected obstacles. In order to test this situation the robot has been located in one of the distinguishable rooms and the normal Gaussian noise has been contaminated with a uniform distribution between the 25% and 75% of the sensor measurement. The results with a 10% of contamination can be observed in Table II (columns with subindex C).

Comparing the results presented in the no contaminated case, there is no clear evidence to choose one option, as in both cases the errors present comparable values. Only the position error is considered, because the results were negligible with the orientation error.

However, if we study the case with uniform contamination, the results are much more interesting. The error estimating the pose using the L1-norm (the average error is equal to 1.087 cm) is significantl lower than using the L2-norm (the average error is 8.464 cm). This is a logical result, because the contamination introduced has a bigger influenc in the penalization of the cost function value using the square error, being much more robust with the firs order one.

Moreover, if we look at the terms in parentheses, which correspond to the population size used in each case, we fin another interesting conclusion. It is very often necessary to use a larger population to succeed (with a smaller size the robot was unable to locate) in cases of contamination when using the L2-norm, which adversely affects the computational cost.

Therefore, we obtain better solutions with the cost function given by the L1-norm in cases where there is a significan contamination (cases with many dynamic objects, pedestrians, and so on).

6.5. *Complexity and computational cost*

A fina issue addressed in our experiments concerns the running time of the RELF-3D algorithm. The computational cost of any algorithm that works in 3D maps is very high, and this is one of the biggest problems found by researchers in this field The absolute time depends on several factors: the computer platform, the observation prediction model and the sensor data, and the population and iterations number.

Table II. L1-norm and L2-norm estimation position errors from different locations. Two different situations: no contamination (NC), and 10% of contamination (C). Errors in centimeters and locations (x, y, z, θ) in cells. Each cell size is a cube of 12-cm side. Population size required is in parentheses.

True Pose	$e_{P(L1,NC)}$	$e_{P(L2,NC)}$	$e_{P(L1,C)}$	$e_{P4(L2,C)}$
(95, 430, 12, 180)	0.783 (200)	1.123 (150)	4.040 (250)	18.84 (200)
(25, 190, 6,0)	0.113 (250)	0.303 (250)	0.570 (250)	1.222 (250)
(50, 100, 10, 0)	0.672 (150)	0.665 (150)	0.656 (150)	0.576 (300)
(50, 350, 13, 0)	0.943 (200)	1.907 (200)	1.646 (200)	3.324 (200)
(90, 425, 8, 270)	0.652 (150)	0.628 (150)	0.689 (150)	12.121 (150)
(80, 240, 15, 270)	0.623 (150)	1.037 (150)	0.907 (150)	4.158 (150)
(70, 350, 8, 270)	0.661 (200)	0.221 (150)	1.605 (150)	1.656 (500)
(30, 225, 14, 90)	0.333 (150)	0.278 (150)	0.151 (150)	13.059 (150)
(30, 200, 5, 90)	0.673 (150)	0.811 (150)	0.366 (150)	13.345 (300)
(25, 275, 10, 90)	0.289 (150)	0.461 (150)	0.246 (150)	16.342 (300)
Average	0.574	0.743	1.087	8.464

Table III. Computation time for a fi ed number of iterations.

Population/iterations	Time (s)
200/15	16.14
100/15	8.77
50/15	4.09
25/15	2.25
10/15	1.04

Table IV. Computation time for a fi ed population of 50 elements.

Population/iterations	Time (s)
50/5	1.42
50/10	2.95
50/15	4.03
50/20	5.66
50/25	7.21
50/50	12.75

The algorithm complexity is $O(n)$, which means that it grows linearly with the population size and the perceptive sensor size. It does not depend on the robot's degrees of freedom, which can be easily expanded if necessary.

Table III shows the time required for RELF-3D to update the estimate value using 9×61-range laser data, 15 iterations/step, and different population sizes. Table IV shows the time required for the RELF-3D algorithm to update the robot's pose estimate using a fi ed population size and changing iterations per step number.

Analyzing the localization method described in Algorithm 2, the bottleneck from a computational point of view is the function named as *dist_est_3d*. It consists of estimating the 3D laser scan (composed of 9×61 laser beams) from a pose estimate in the simulated environment, and it has to be calculated for each population member. The reading is used by the fitnes function to calculate the error.

A critical point in any global localization algorithm is the variation in the computational requirements with environment dimensions. The RELF-3D algorithm requirements in our experiments not only depend on the size of the environment but also on the characteristics of the place where the robot is located. As an example, in the experiments done in our laboratory test site, a 900 $m^2 \times 3$ m environment, the algorithm requires at least 50 samples when it is in the corridor, and 100 when it is in one of the offices

The computational cost is higher when we use the algorithm in three dimensions, but the improvement of the accuracy is an important result. If we work with a mobile manipulator and tasks with small objects, it will be indispensable to obtain an accuracy of millimeters. If the robot does not need this level of accuracy, we will obtain a reduction in computational cost with less restrictive convergence conditions.

It is not easy to compare our algorithm to other methods. Se *et al.*'s[22] global localization computational times are 0.725 s and 0.02 s for two different methods implemented. They show that the SIFT feature extraction is efficien from a computational point of view. However, their method is vision-based and the computational problems that they need to solve are different. Besides, they do not focus on obtaining the maximum accuracy and their map size is 100 m^2 (our map size is 900 m^2). Ho *et al.*'s[8] algorithm needs 4.82 s to obtain the global localization solution. This time is similar to the time obtained in our results. Lingemann *et al.*'s[13] method can be used online because they need 10.38 s to compute a trajectory composed of 2844 scans. However, it is not easy to compare their method to our algorithm because their times are referred to the tracking problem. Finally, the time needed to obtain measurements can be the most important one in many cases and it has to be considered in real applications. The time delays of ultrasonic transmitter/receiver modules are approximately equal to 0.5 s in the paper published by Tsai *et al.*[27]

7. Conclusions

The last version of our global localization method based on evolutionary concepts is presented in this paper. The ability of the algorithm to successfully perform its tasks is demonstrated, and the influenc of various improvements, such as thresholding and discarding, is also discussed.

As we have demonstrated by the results obtained in the tests carried out, this new version is able to improve the accuracy in an order of magnitude in comparison with the

results obtained with the original method in 2D maps. The increase in the amount of information allows us to obtain an accuracy of millimeters, which is below the accuracy of the laser manipulation tasks with a mobile manipulator. This feature is one of the most important conclusions of the presented work.

It is necessary to emphasize that the increment in the computational cost due to the increase in the amount of handled information can be limited, relaxing the accuracy and convergence requirements. The computational cost is an important limitation to be addressed because it is a common problem in this type of methods. It can be seen in the previous section where the computational cost is compared with those obtained by other methods. Besides, the population reduction after convergence makes it possible to use the algorithm online.

Due to the stochastic nature of the search algorithm for the robot's best pose estimate, the algorithm is able to cope with a high level of sensor noise with low degradation of estimation results.

A thresholding mechanism has been also implemented. The consequence is the capability of the RELF-3D to avoid the premature convergence toward some areas, not selecting the new population members that can be originated by the noise. The introduction of a threshold level has an important disadvantage: the decrease of the convergence speed. It was therefore necessary to develop a discarding mechanism to improve the convergence speed while maintaining the positive aspects of thresholding. The convergence speed after discarding is even higher than with the original method.

A further study to compare the L1-norm and the L2-norm as a cost function has been done in this work. We can conclude that with a Gaussian noise the behavior is similar in both cases, while the use of the first-orde error is useful in certain situations, such as those where there are dynamic objects, people, outliers, and so on.

In addition, the developed algorithm has many other characteristics that make its use very interesting: It can deal with nonlinear state space dynamics and noise distributions, it does not require any assumptions on the shape of the posterior density, the computational resources focus on the most relevant areas, and so on.

References

1. M. Agrawal and K. Konolige, "Real-Time Localization in Outdoor Environments using Stereo Vision and Inexpensive Gps," *Proceedings of the International Conference on Pattern Recognition (ICPR'06)* Hong Kong (Aug 20–24, 2006).
2. W. Burgard, D. Fox, D. Henning and T. Schmidt, "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids," *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, Oregon (Aug 4–8, 1996).
3. D. M. Cole and P. M. Newman, "Using Laser Range Data for 3D Slam in Outdoor Environments," *Proccedings of IEEE International Conference on Robotics and Automation (ICRA'06)*, Orlando, Florida (May 2006).
4. A. Diosi and L. Kleeman, "Laser Scan Matching in Polar Coordinates with Application to Slam," *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'05)*, Alberta, Canada (Aug 2–6, 2005).
5. A. Doucet, "On Squential Simulation-Based Methods for Bayesian Filtering," Technical Report, CUED/FINFENG/TR 31 (Department of Engineering, Cambridge University, Cambridge, UK, 1998).
6. D. Fox, W. Burgard and S. Thrun, "Markov localization for mobile robots in dynamic environments," *J. Artif. Intell. Res.* **11**(11), 391–427 (1999).
7. D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning* (Addison Wesley, 1989).
8. N. Ho and R. Jarvis, "Vision-Based Global Localisation Using a 3D Environmental Model Created by a Laser Range Scanner," *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'08)*, Acropolis Convention Center, Nice, France (Sep. 2008).
9. P. Jensfelt, "Approaches to mobile robot localization in indoor environments" *PhD thesis* (Royal Institute of Technology, Sweden, 2001).
10. T. Krink, B. Filipic and G. B. Fogel, "Noisy Optimization Problems – a Particular Challenge for Differential Evolution?," *Proceedings of the Congress on Evolutionary Computation (CEC'04)*, San Diego, California (July 6–9, 2004).
11. R. Kümmerle, R. Triebel, P. Pfaff and W. Burgard, "Monte Carlo localization in outdoor terrains using multi-level surface maps," *J. Field Robot.* **42**, 213–222 (2008).
12. L. C. Lai, T. L. Lee, H. T. Fan and C. J. Wu, "A Nonlinear Programming Method for 3d Localization of Mobile Robots," *International Conference on Advanced Robotics*, Seattle, Washington (Jul. 18–20, 2005).
13. K. Lingemann, A. Nüchter, J. Hertzberg and H. Surmann, "High-speed laser localization for mobile robots," *Robot. Auton. Syst.* **51**, 275–296 (2005).
14. D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," *Seventh International Conference on Computer Vision (ICCV'99)*, Kerkyra, Corfu, Greece (Sep. 20–25, 1999).
15. S. Markon, D. V. Arnold, T. Back, T. Beielstein and H.-G. Beyer, "Thresholding-a Selection Operator for Noisy Es," *Proceedings of the Congress on Evolutionary Computation (CEC'01)*, Seoul, Korea (2001).
16. F. Martín, C. González, L. Moreno and D. Blanco, "Accelerated Localization in Noisy 3D Environments Using Differential Evolution," *Proceedings of International Conference on Genetic and Evolutionary Methods*, Las Vegas, Nevada (Jul. 12–15, 2010).
17. F. Martín, L. Moreno, S. Garrido and D. Blanco, "Localization in 3D Environments Using Differential Evolution," *International Symposium on Intelligent Signal Processing (WISP'2009)*, Budapest, Hungary (Aug. 26–28, 2009).
18. F. Martín, M. L. Muñoz, S. Garrido, D. Blanco and L. Moreno, "L1-norm Global Localization Based on a Differential Evolution Filter," *International Symposium on Intelligent Signal Processing (WISP'2009)*, Budapest, Hungary (Aug. 26–28, 2009).
19. L. Moreno, S. Garrido and M. L. Muñoz, "Evolutionary filte for robust mobile robot localization," *Robot. Auton. Syst.* **54**(7), 590–600 (2006).
20. A. Nüchter, K. Lingemann and J. Hertzberg, "6D slam – 3D mapping outdoor environments," *J. Field Robot.* **24**, 699–722 (2007).
21. E. Royer, M. Lhuillier, M. Dhome and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *Int. J. Comput. Vis.* **74**, 237–260 (2007).
22. S. Se, D. G. Lowe and J. J. Little, "Vision-based global localization and mapping for mobile robots," *IEEE Trans. Robot.* **21**, 3 (2005).
23. M. Sridharan, G. Kuhlmann and P. Stone, "Practical Vision-Based Monte Carlo Localization on a Legged Robot," *International Conference on Robotics and Automation*

(ICRA'05), Barcelona, Spain, vol. 3 (Apr. 18–22, 2005) pp. 3366.
24. R. Storn and K. Price, "Differential evolution – a simple and efficien Heuristic for global optimization over continuous spaces," *J. of Glob. Optim.* **11**, 341–359 (Dec. 1997).
25. S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer and W. Whittaker, "A System for Volumetric Robotic Mapping of Abandoned Mines," *Proceedings of the International Conference on Robotics and Automation (ICRA'03)* (May 12–17, 2003).
26. R. Triebel, P. Pfaff and W. Burgard, "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'06)*, Beijing, China (Oct. 2006).
27. C.-C. Tsai H.-H. Lin and S.-W. Lai, "Multisensor 3D posture determination of a mobile robot using inertial and ultrasonic sensors," *J. Intell. Robot. Syst.* **42**, 317–335 (2005).
28. A. R. Vahdat, N. N. Ashrafoddin and S. S. Ghidary, "Mobile Robot Global Localization using Differential Evolution and Particle Swarm Optimization," *Proceedings of the Congress on Evolutionary Computation (CEC'07)* (Sep. 25–28, 2007).
29. J. Wolf, W. Burgard and H. Burkhardt, "Robust vision-based localization by combining an image retrieval system with monte carlo localization," *IEEE Trans. Robot.* **21**, 2 (2005).